Government Engineering College

Sec-28 Gandhinagar

Sem:-V (Computer Engineering Department)

Subject: Python with Data Science [3150713]

Certificate

This is to certify that

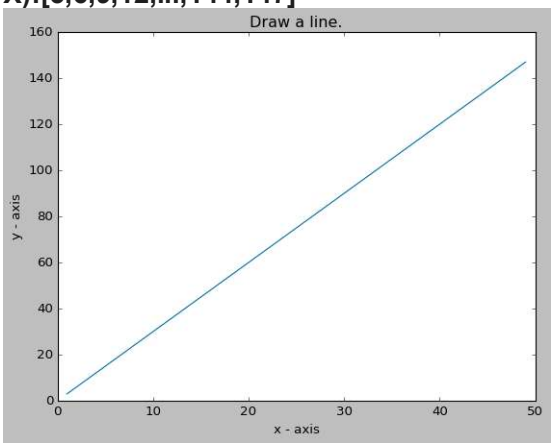
Mr./~~Ms~~ **NIRAJKUMAR ITALIYA** Of class **CE_Sem-5** Division **A2** Enrollment No **190130107041** Has satisfactorily completed his/~~her~~ term work in **PDS(*3150713)*** Subjec t for the term ending in November **2021-2022.**

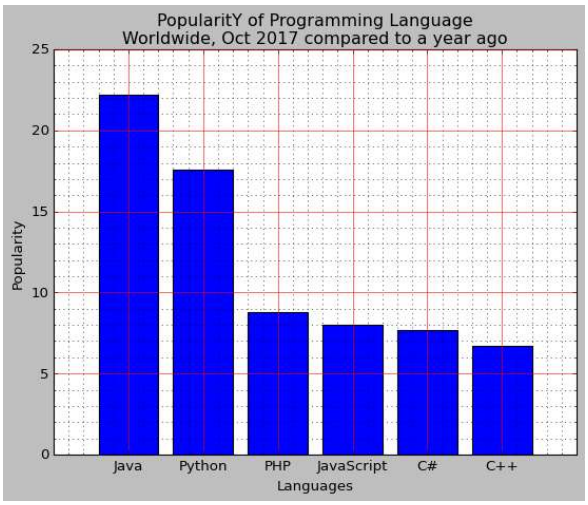
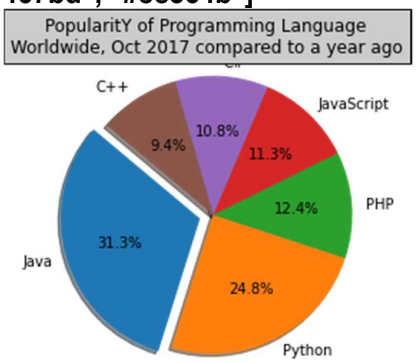
Date:- **27/09/2021**

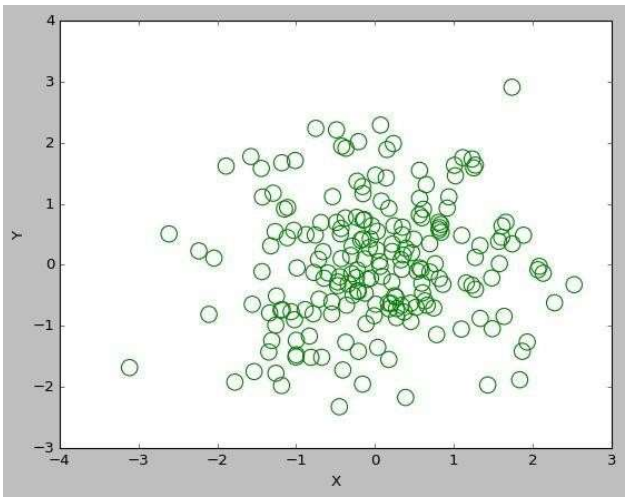
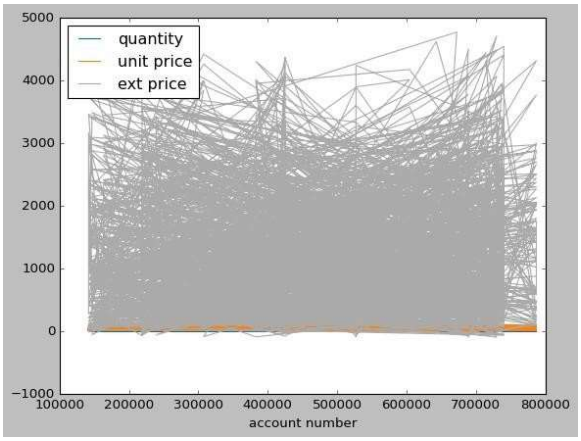
Practicals

No.	Basic Python Practicals		Date	Page No.	Sign
1	**Develop a program to understand the control structures of python.**	CO1	31/07/2021		
2	**Develop a program to learn different types of structures (list, dictionary, tuples) in python.**	CO1	31/07/2021		
3	**Develop a program to learn concepts of functions scoping, recursion and list mutability.**	CO1	31/07/2021		
4	**Develop a program to understand working of exception handling and assertions.**	CO1	31/07/2021		
5	**Develop a program for data structure algorithms using python – searching, sorting, and hash tables.**	CO1	31/07/2021		
6	**Develop a program to learn regular expressions using python.**	CO1	31/07/2021		
7	**Develop chat room applications using multithreading.(Assignment)**	CO1	31/07/2021		
8	**Learn to plot different types of graphs using PyPlot.**	CO1	31/07/2021		
9	**Implement classical ciphers using python. .(Assignment)**	CO1	31/07/2021		
10	**Draw graphics using Turtle. .(Assignment)**	CO1	31/07/2021		
11	**Develop a program to learn GUI programming using Tkinter. .(Assignment)**	CO1	31/07/2021		
	Pandas Library Practicals				
12	**Develop a program that reads a .csv dataset file using Pandas library and display the following content of the dataset. a) First five rows of the dataset b) Complete data of the dataset c) Summary or metadata of the dataset**	CO2	30/08/2021		
13	**Develop a program that shows application of slicing and dicing over the raws and columns of the dataset.**	CO2	30/08/2021		
14	**Develop a program that shows usage of aggregate function over the input dataset. a) describe b) max c) min d) mean e) median f) count g) std h) Corr**	CO2	30/08/2021		

15	**Develop a program that applies split and merge operations on the datasets.**	CO2	30/08/2021		
16	**Develop a program that shows the various data cleaning tasks over the dataset.** **a) Identifying the null values** **b) Identifying the empty values** **c) Identifying the incorrect timestamp**	CO3	30/08/2021		
17	**Develop a program that shows an application of a Lamda function over the dataset.**	CO2	30/08/2021		
	NumPy Library Practicals				
18	**Develop a program that shows usage of following NumPy array operations:** **a) any()** **b) all()** **c) isnan()** **d) isinf()** **e) isfinite()** **f) isinf()** **g) zeros()** **greater_equal()** **h) isreal()** **i) iscomplex()** **j) isscalar()** **k) less()** **l) greater()** **m) less_equal()** **o)**	CO3	10/09/2021		
19	**Develop a program that shows usage of following NumPy library vector functions.** **a) arrange()** **b) reshape()** **c) linspace()** **d) randint()** **e) dot()**	CO3	10/09/2021		
	Matplotlib Library Practicals				
20	**Write a program to display below plot using matplotlib library**				

	For Values of X:[1,2,3,...,49], Values of Y (thrice of X):[3,6,9,12,...,144,147] 	CO4	10/09/2020	66-6	
21	**Write a program to display below plot using matplotlib library For the point values** **x1 = [10,20,30], y1 = [20,40,10]** **x2 = [10,20,30], y2 = [40,10,30]**	CO4	10/09/2020	67-67	
22	**Write a program to display below bar plot using matplotlib library** **For value** **Languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++'] popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]**	CO4	10/09/2020	68-69	

					
23	**Write a program to display below bar plot using matplotlib library** **For below data display pie plot** **languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']** **popuratity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]** **colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b"]** 	CO4	10/09/2020	70-70	
24	**Write a program to display below bar plot using matplotlib library** **For 200 random points for both X and Y display scatter plot:**	CO4	10/09/2020	71-71	

					
25	**Develop a program that reads .csv file from the url: (https://github.com/chris1610/pbpython/blob/master/data/sample-salesv3.xlsx?raw=true) and plot the data of the dataset stored in the .csv file.** 	CO4	10/09/2020	72-72	
	Scikit Learn Practicals				
26	**Exercise 1: Language identification** ● **Write a text classification pipeline using a custom preprocessor and CharNGramAnalyzer using data from Wikipedia articles as a training set.** ● **Evaluate the performance on some held out test sets.**	CO5	15/10/2020	73-73	
27	**Exercise 2: Sentiment Analysis on movie reviews** ● **Write a text classification pipeline to classify movie reviews as either positive or negative.** ● **Find a good set of parameters using grid search.**	CO5	15/10/2020	74-75	

	• **Evaluate the performance on a held out test set.**				
28	**CLI text classification utility** **Using the results of the previous exercises and the cPickle module of the standard library, write a command line utility that detects the language of some text provided on stdin and estimates the polarity (positive or negative) if the text is written in English.** **Bonus point if the utility is able to give a confidence level for its predictions.**	CO5	15/10/2020	76-77	

Practicals

enroll = 190130107041

1) Develop a program to understand the control structures of python.

```
print("Enrollment no =",enroll)
print("Practical no= 1")
for i in range(5):
    if i%2 == 0:
        print(str(i) + ' Even')
    elif i%3==0:
        print(str(i) + ' divisible by 3')
    else:
        print(str(i) + ' odd')
i=0
while i < 3:
    print("I'm loopy!")
    i += 1
    if i == 1:
        print('Positive value')
    else:
        print("Negative value")
```

Output :

```
print("Enrollment no =",enroll)
print("Practical no= 1")
for i in range(5):
    if i%2 == 0:
        print(str(i) + ' Even')
    elif i%3==0:
        print(str(i) + ' divisible by 3')
    else:
        print(str(i) + ' odd')
i=0
while i < 3:
    print("I'm loopy!")
    i += 1
    if i == 1:
        print('Positive value')
    else:
        print("Negative value")
```

```
Enrollment no = 190130107041
Practical no= 1
0 Even
1 odd
2 Even
3 divisible by 3
4 Even
I'm loopy!
Positive value
I'm loopy!
Negative value
I'm loopy!
Negative value
```

2) Develop a program to learn different types of structures (list, dictionary, tuples) in python.

```
print("Enrollment no =",enroll)
print("Practical no= 2")
print("Practical Title= understand different types of structure(list, dictionary, tuples)")
lst = list()

for i in range(1, 6):
    lst.append(i)
    print('Our list:', lst)
lst[2] = 'PYTHON'

print('after mutation:', lst)
dct = {}
fruit = ['apple', 'banana', 'grapes']
for i in range(1, 4):
    dct[i] = fruit[i-1]
    print('dict:', dct)
    tpl = ([ft for ft in fruit])
    print(tpl)
A = set([1, 2, 3, 'a', 1, 'b'])

print(A)
```

Output :

practical 2

```
print("Enrollment no =",enroll)
print("Practical no= 2")
print("Practical Title= understand different types of structure(list, dictionary, tuples)")
lst = list()

for i in range(1, 6):
    lst.append(i)
    print('Our list:', lst)
lst[2] = 'PYTHON'

print('after mutation:', lst)
dct = {}
fruit = ['apple', 'banana', 'grapes']
for i in range(1, 4):
    dct[i] = fruit[i-1]
    print('dict:', dct)
    tpl = ([ft for ft in fruit])
    print(tpl)
A = set([1, 2, 3, 'a', 1, 'b'])

print(A)
```

```
Enrollment no = 190130107041
Practical no= 2
Practical Title= understand different types of structure(list, dictionary, tuples)
Our list: [1]
Our list: [1, 2]
Our list: [1, 2, 3]
Our list: [1, 2, 3, 4]
Our list: [1, 2, 3, 4, 5]
after mutation: [1, 2, 'PYTHON', 4, 5]
dict: {1: 'apple'}
['apple', 'banana', 'grapes']
dict: {1: 'apple', 2: 'banana'}
['apple', 'banana', 'grapes']
dict: {1: 'apple', 2: 'banana', 3: 'grapes'}
['apple', 'banana', 'grapes']
{1, 2, 'b', 3, 'a'}
```

3) Develop a program to learn concepts of functions scoping, recursion and list mutability.

```
print("Enrollment no =",enroll)
print("Practical no= 3")
print("Practical Title= understand the functions scoping, recursion and list mutabilit")
import datetime as dt
def myfunc():
    x=100
print(x)
myfunc()
def get_date():
    dt = datetime.datetime.now().date()
def printfnc():
    print("Today's date: " + str(dt))
printfnc()
get_date()


def my_name():
    name = 'Joey'
    print(name)
    name='Joey Tribbiani'
my_name()
def get_itr(num):
    if num <= 0: return 1
    elif num == 1: return 1
    else:
        return num*get_itr(num-1)
ans = get_itr(5)
print(ans) #list
lst = [i*i for i in range(1, 6)]
print('Before mutation: ' + str(lst[2]))
```

lst[2] = '1024'

print('After mutation: ' + str(lst[2]))

Output:

```
In [53]: print("Enrollment no =",enroll)
         print("Practical no= 3")
         print("Practical Title= understand the functions scoping, recursion and list mutabilit")
         import datetime as dt
         def myfunc():
             x=100

         print(x)
         myfunc()
         def get_date():
             dt = datetime.datetime.now().date()
         def printfnc():
             print("Today's date: " + str(dt))
         printfnc()
         get_date()



         def my_name():
             name = 'Joey'
             print(name)
             name='Joey Tribbiani'
         my_name()
         def get_itr(num):
             if num <= 0: return 1
             elif num == 1: return 1
             else:
                 return num*get_itr(num-1)
         ans = get_itr(5)
         print(ans) #list
         lst = [i*i for i in range(1, 6)]
         print('Before mutation: ' + str(lst[2]))
         lst[2] = '1024'
         print('After mutation: ' + str(lst[2]))

         Enrollment no = 190130107041
         Practical no= 3
         Practical Title= understand the functions scoping, recursion and list mutabilit
         200
         Today's date: <module 'datetime' from 'C:\\Users\\Mahadev\\anaconda3\\lib\\datetime.py'>
         Joey
         120
         Before mutation: 9
         After mutation: 1024
```

Develop a program to understand working of exception handling and assertions. Practical 4.1:

```
print("Enrollment no
=",enroll) print("Practical
no= 4.1")
print("Practical Title= understand the working of
assertion") #P4_1
#assertion
def convert1(Temp):
   assert (Temp >= 0), "Colder than absolute
   zero!" return ((Temp-273)*1.8)+32
#print (convert1(273))
print
(int(convert1(505.78)))
#print (convert1(-5))
```

Output:

```
print("Enrollment no =",enroll)
print("Practical no= 4.1")
print("Practical Title= understand the working of assertion") #P4_1
#assertion

def convert1(Temp):

    assert (Temp >= 0), "Colder than absolute zero!"
    return ((Temp-273)*1.8)+32


print (int(convert1(505.78)))
```

```
Enrollment no = 190130107041
Practical no= 4.1
Practical Title= understand the working of assertion
451
```

Practical 4.2:

```
print("Enrollment no
=",enroll) print("Practical
no= 4.2")
print("Practical Title= understand the excaption
handling") #P4_2
a = 100
b = int(input('Enter b: '))
try:
   ans =
a/b
except:
   print('invalid b
value!') else:
   print('Ans: ' +
str(ans)) finally:
   print('This is a demo of try and except.')
```

Output :

```
print("Enrollment no =",enroll)
print("Practical no= 4.2")
print("Practical Title= understand the excaption handling") #P4_2
a = 100

b = int(input('Enter b: '))
try:
    ans = a/b
except:
    print('invalid b value!')
else:
    print('Ans: ' + str(ans))
finally:
    print('This is a demo of try and except.')
```

```
Enrollment no = 190130107041
Practical no= 4.2
Practical Title= understand the excaption handling
Enter b: 5
Ans: 20.0
This is a demo of try and except.
```

5) Develop a program for data structure algorithms using python – searching, sorting, and hash tables.

```
print("Enrollment no
=",enroll) print("Practical
no= 5.1")
print("Practical Title= insertion sort
algorithm ") #P5_1
def insertionSort(arr):
   for i in
      range(1,len(arr)):
      key = arr[i]
      j = i-1
   while j >= 0 and key <
      arr[j] : arr[j+1] = arr[j]
      j-= 1
   arr[j+1] = key
arr = ['w', 'e', 'l', 'c', 'o',
'm','e'] print("Main String is:
") print(arr)
insertionSort(arr)
print("Sorted string
is:") print(sorted(arr))
```

Output:

```
print("Enrollment no =",enroll)
print("Practical no= 5.1")
print("Practical Title= insertion sort algorithm ") #P5_1
def insertionSort(arr):
    for i in range(1,len(arr)):
        key = arr[i]
    j = i-1
    while j >= 0 and key < arr[j] :
        arr[j+1] = arr[j]
        j-= 1
    arr[j+1] = key

arr = ['w', 'e', 'l', 'c', 'o', 'm','e']
print("Main String is: ")
print(arr)
insertionSort(arr)
print("Sorted string is:")
print(sorted(arr))
```

```
Enrollment no = 190130107041
Practical no= 5.1
Practical Title= insertion sort algorithm
Main String is:
['w', 'e', 'l', 'c', 'o', 'm', 'e']
Sorted string is:
['c', 'e', 'e', 'l', 'm', 'o', 'w']
```

Practical 5.2:

```
print("Enrollment no
=",enroll) print("Practical
no= 5.2")
print("Practical Title= Mergesort
algorithm") #P5_2
def merge(arr, l, m,
  r): n1 = m - l + 1
  n2 = r- m
  L = [0] * (n1)
  R = [0] * (n2)
  for i in range(0 ,
    n1): L[i] = arr[l +
    i]
  for j in range(0 ,
    n2): R[j] = arr[m
    + 1 + j]
  i = 0
  j = 0
  k = l
  while i < n1 and j <
    n2 : if L[i] <= R[j]:
      arr[k] =
      L[i] i += 1
    else:
      arr[k] =
      R[j] j += 1
    k += 1
  while i < n1:
    arr[k] =
    L[i]
```

```
        i += 1
        k += 1
    while j < n2:
        arr[k] =
        R[j] j += 1
        k += 1
def mergeSort(arr,l,r):
    if l < r:
        m = (l+(r-1))//2
        mergeSort(arr, l, m)
        mergeSort(arr, m+1,
        r) merge(arr, l, m, r)
arr = [14,46,43,27,57,41,45,21,70]
n = len(arr)
print('Unsorted list:')
print(arr)
mergeSort(arr, 0, n-
1) print('Sorted list:
') print(arr)
```

Output :

```
print("Enrollment no =",enroll)
print("Practical no= 5.2")
print("Practical Title= Mergesort algorithm") #P5_2
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r- m
    L = [0] * (n1)
    R = [0] * (n2)

    for i in range(0 , n1):
        L[i] = arr[l + i]
    for j in range(0 , n2):
        R[j] = arr[m + 1 + j]
    i = 0
    j = 0
    k = l
    while i < n1 and j < n2 :
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

def mergeSort(arr,l,r):
    if l < r:
        m = (l+(r-1))//2
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)

arr = [14,46,43,27,57,41,45,21,70]

n = len(arr)
print('Unsorted list:')
print(arr)
mergeSort(arr, 0, n-1)
print('Sorted list: ')
print(arr)
```

```
Enrollment no = 190130107041
Practical no= 5.2
Practical Title= Mergesort algorithm
Unsorted list:
[14, 46, 43, 27, 57, 41, 45, 21, 70]
Sorted list:
[14, 21, 27, 41, 43, 45, 46, 57, 70]
```

6) Develop a program to learn regular expressions using python.

```
print("Enrollment no
=",enroll) print("Practical
no= 6")
print("Practical Title= understand the regular
expression") #P6
import re
text = input('Enter string/paragraph: ')
pattern = input('Enter the pattern to find:
') if re.search(pattern, text):
   print('substring
found') else:
   print('substring not
found')
```

Output :

```
print("Enrollment no =",enroll)
print("Practical no= 6")
print("Practical Title= understand the regular expression") #P6
import re
text = input('Enter string/paragraph: ')
pattern = input('Enter the pattern to find: ')
if re.search(pattern, text):
      print('substring found')
else:
    print('substring not found')
```

```
Enrollment no = 190130107041
Practical no= 6
Practical Title= understand the regular expression
Enter string/paragraph: hi I am Niraj Italiya
Enter the pattern to find: Nira
substring found
```

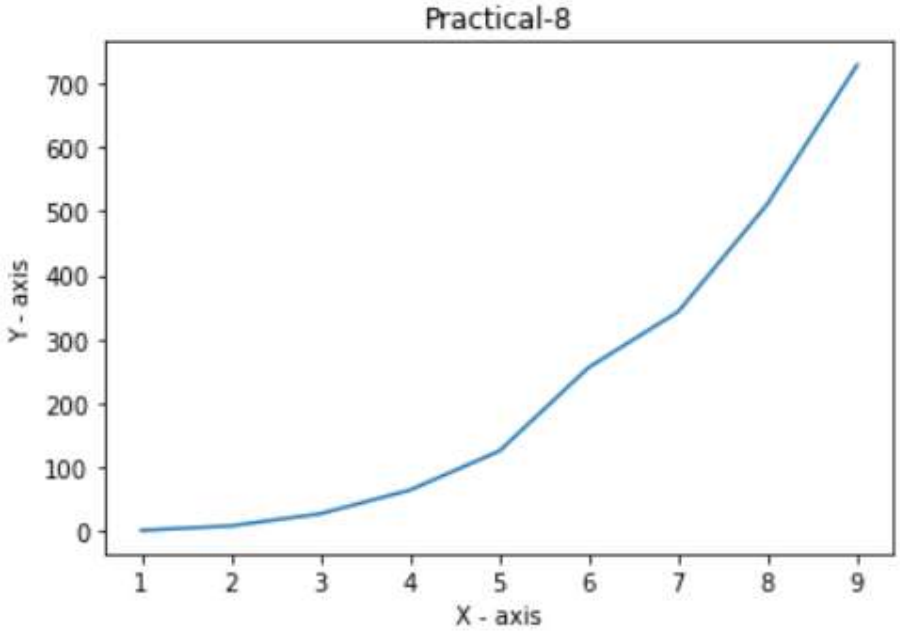
7) Learn to plot different types of graphs using PyPlot.

```
print("Enrollment no =",enroll)
print("Practical no= 8")
print("Practical Title= learn different types of graphs ") #P8_1

import matplotlib.pyplot as plt
x = [1,2,3,4,5,6,7,8,9]
y = [1,8,27,64,125,256,343,512,729]

plt.plot(x, y)
plt.xlabel('X - axis')
plt.ylabel('Y - axis')
plt.title('Practical-8')
plt.show()
```

Output:

```
print("Enrollment no =",enroll)
print("Practical no= 8")
print("Practical Title= learn different types of graphs ") #P8_1

import matplotlib.pyplot as plt
x = [1,2,3,4,5,6,7,8,9]
y = [1,8,27,64,125,256,343,512,729]

plt.plot(x, y)
plt.xlabel('X - axis')
plt.ylabel('Y - axis')
plt.title('Practical-8')
plt.show()
```

```
Enrollment no = 190130107041
Practical no= 8
Practical Title= learn different types of graphs
```



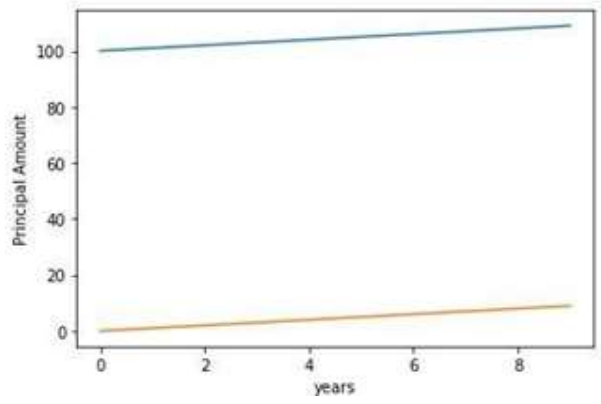
Practical 8.2:

```
print("Enrollment no =",enroll)
print("Practical no = 8.2")
print("Practical Title= learn different types of graphs ") #P8_2
#import matplotlib.pyplot as plt

def compound_interest(principle, rate, time):
    result = principle * (pow((1 + rate / 100), time))
    return result

p = float(input("Enter the principal amount: "))
r = float(input("Enter the interest rate: "))
endyear = float(input("Enter the Year : "))

yearlist = []
pamountlist = []
interestlist = []

for i in range(int(endyear)):
    yearlist.append(i)

amount = compound_interest(p, r, i)
intamount = int(amount)
pamountlist.append(intamount)
interest = amount - p
intins = int(interest)
interestlist.append(intins)

print(yearlist)
print(pamountlist)
print(interestlist)

plt.plot(yearlist,pamountlist,interestlist)

plt.xlabel('years')
plt.ylabel('Principal Amount')
plt.show()
```

Output:

```
print("Enrollment no =",enroll)
print("Practical no = 8.2")
print("Practical Title= learn different types of graphs ") #P8_2
#import matplotlib.pyplot as plt

def compound_interest(principle, rate, time):
    result = principle * (pow((1 + rate / 100), time))
    return result

p = float(input("Enter the principal amount: "))
r = float(input("Enter the interest rate: "))
endyear = float(input("Enter the Year : "))

yearlist = []
pamountlist = []
interestlist = []

for i in range(int(endyear)):
    yearlist.append(i)

amount = compound_interest(p, r, i)
intamount = int(amount)
pamountlist.append(intamount)
interest = amount - p
intins = int(interest)
interestlist.append(intins)

print(yearlist)
print(pamountlist)
print(interestlist)

plt.plot(yearlist,pamountlist,interestlist)

plt.xlabel('years')
plt.ylabel('Principal Amount')
plt.show()
```

```
Enrollment no = 190130107041
Practical no = 8.2
Practical Title= learn different types of graphs    phs
Enter the principal amount: 100
Enter the interest rate: 1
Enter the Year : 10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[100, 101, 102, 103, 104, 105, 106, 107, 108, 109]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



Pandas library practicals:

Develop a program that reads a .csv dataset file using Pandas library and display the following content of the dataset.

```
import pandas as pd
```

```
data = iris = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
```

8) Develop a program that reads a .csv dataset file using Pandas library and display the following content of the dataset.

a) First five rows of the dataset

```
data.head()
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

b) Complete data of the dataset

```
data
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

c) Summary or metadata of the dataset

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

9) Develop a program that shows application of slicing and dicing over the raws and columns of the dataset.

```
import pandas as pd
import numpy as np
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.loc[:,'A'])
```

```
Enrollment no = 190130107041
a    0.758860
b   -1.239185
c    0.306514
d   -0.060330
o   -0.433738
p    0.521186
r   -1.442198
s    1.004089
Name: A, dtype: float64
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.loc[:,['A','C']])
```

```
Enrollment no = 190130107041
          A         C
a -1.148502 -0.926495
b  0.346518  0.890746
c -0.724899 -1.078188
d -0.634374  0.237977
o  1.326304 -0.371999
p  0.178091 -1.458619
r  0.612559  0.607555
s -0.186330 -0.643707
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.loc[['a','b','p','r'],['A','C']])
```

```
Enrollment no = 190130107041
          A         C
a -0.552819  0.493759
b -0.972543 -0.502585
p -0.066974 -0.197435
r  0.259349  0.615152
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.loc['a' : 'r'])
```

```
Enrollment no = 190130107041
          A         B         C         D
a -1.980180  0.695717  1.007157  0.486581
b  0.698686  1.227996  0.850285  0.298570
c  0.190421  1.837260  0.207050  1.205270
d  0.589866  0.071503  1.288403 -0.884647
o  0.054796 -1.292724  0.344578  0.178822
p  0.140658  0.088828 -1.660217 -0.680578
r  0.174287  0.299880 -0.515118 -1.981238
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.loc['a' ] > 0)
```

```
Enrollment no = 190130107041
A     True
B     True
C     True
D    False
Name: a, dtype: bool
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.iloc[:4])
```

```
Enrollment no = 190130107041
          A         B         C         D
a  0.095080  0.493221  0.084660  0.237540
b -1.174525  0.945795  0.262014 -0.393718
c -1.476485  0.359280 -0.602045  1.489333
d -1.405216  1.150499  0.017048  0.186504
```

```
print("Enrollment no =",enroll)
df = pd.DataFrame(np.random.randn(8,4),
                  index = ['a','b','c','d','o','p','r','s'],
                  columns = ['A','B','C','D'])
print(df.iloc[:4])
print(df.iloc[1:5,2:4])
```

```
Enrollment no = 190130107041
          A         B         C         D
a  0.682674  0.711788  0.167397 -0.976270
b  0.232862 -1.681767 -1.325783 -0.058542
c -2.587614  0.285982 -0.837404  0.479718
d -1.146150 -0.591694 -0.763385  1.249983
          C         D
b -1.325783 -0.058542
c -0.837404  0.479718
d -0.763385  1.249983
o  0.736802 -0.544630
```

10) Develop a program that shows usage of aggregate function over the input dataset.

a) describe

```
print("Enrollment no =",enroll)
data.describe()
```

```
Enrollment no = 190130107041
```

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

b) max

```
print("Enrollment no =",enroll)
data.max()
```

```
Enrollment no = 190130107041

5.1                         7.9
3.5                         4.4
1.4                         6.9
0.2                         2.5
Iris-setosa    Iris-virginica
dtype: object
```

c) min

```
print("Enrollment no =",enroll)
data.min()
```

```
Enrollment no = 190130107041

5.1                    4.3
3.5                      2
1.4                      1
0.2                    0.1
Iris-setosa    Iris-setosa
dtype: object
```

d) mean

```
print("Enrollment no =",enroll)
data.iloc[100:151].mean()
```

```
Enrollment no = 190130107041

5.1    6.593878
3.5    2.967347
1.4    5.542857
0.2    2.016327
dtype: float64
```

e) median

```
print("Enrollment no =",enroll)
data.iloc[100:151].median()
```

```
Enrollment no = 190130107041

5.1    6.5
3.5    3.0
1.4    5.5
0.2    2.0
dtype: float64
```

f) count

```
print("Enrollment no =",enroll)
data.count()
```

```
Enrollment no = 190130107041

5.1             149
3.5             149
1.4             149
0.2             149
Iris-setosa     149
dtype: int64
```

g) std

```
print("Enrollment no =",enroll)
data.std()
```

```
Enrollment no = 190130107041

5.1     0.828594
3.5     0.433499
1.4     1.759651
0.2     0.761292
dtype: float64
```

h) corr

```
print("Enrollment no =",enroll)
data.corr()
```

```
Enrollment no = 190130107041
```

	5.1	3.5	1.4	0.2
5.1	1.000000	-0.103784	0.871283	0.816971
3.5	-0.103784	1.000000	-0.415218	-0.350733
1.4	0.871283	-0.415218	1.000000	0.962314
0.2	0.816971	-0.350733	0.962314	1.000000

11) Develop a program that applies split and merge operations on the datasets.

```
df=[['Braund, Mr.Owen Harris',80,177.0],['Heikkinen, Miss. Laina',78,180.0],['Monvila, Rev.Juozas',87,165.0]]
df2=pd.DataFrame(df,columns=['name','other_disease','Height'])
df2
```

	name	other_disease	Height
0	Braund, Mr.Owen Harris	80	177.0
1	Heikkinen, Miss. Laina	78	180.0
2	Monvila, Rev.Juozas	87	165.0

```
df3=pd.merge(data,df2,how='right',on='other_disease')
df3
```

	id	sex	inmsupr	hypertension	other_disease	cardiovascular	obesity	renal_chronic	tobacco	contact_other_covid	covid_res	icu	name	Height
0	NaN	NaN	NaN	NaN	80	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Braund, Mr.Owen Harris	177.0
1	NaN	NaN	NaN	NaN	78	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Heikkinen, Miss. Laina	180.0
2	NaN	NaN	NaN	NaN	87	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Monvila, Rev.Juozas	165.0

12) Develop a program that shows the various data cleaning tasks over the dataset.

a) Identifying the null values

```
print("Enrollment no =",enroll)
data.isnull().sum()
```

```
Enrollment no = 190130107041

5.1             0
3.5             0
1.4             0
0.2             0
Iris-setosa     0
dtype: int64
```

b) Identifying the empty values

```
print("Enrollment no =",enroll)
data.empty
```

```
Enrollment no = 190130107041

False
```

c) Identifying the incorrect timestamp

```
dffc=pd.to_datetime(df['entry_date'],errors='coerce')
dffc
```

```
0         2020-04-05
1         2020-03-19
2         2020-06-04
3         2020-04-17
4         2020-04-13
             ...
566597    2020-05-13
566598    2020-07-04
566599    2020-05-14
566600    2020-05-31
566601    2020-05-16
Name: entry_date, Length: 566602, dtype: datetime64[ns]
```

13) Develop a program that shows an application of a Lamda function over the dataset.

```
df3['other_disease']=df3['other_disease'].apply(lambda x: other_disease(x))
df3
```

	id	sex	inmsupr	hypertension	other_disease	cardiovascular	obesity	renal_chronic	tobacco	contact_other_covid	covid_res	icu	name	Height
0	NaN	NaN	NaN	NaN	3rd class	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Braund, Mr.Owen Harris	177.0
1	NaN	NaN	NaN	NaN	78	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Heikkinen, Miss. Laina	180.0
2	NaN	NaN	NaN	NaN	87	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Monvila, Rev.Juozas	165.0

Numpy practicals:

14) Develop a programme that shows usage of following NumPy array operations:

```
In [1]: import numpy as np
```

```
print("Enrollment no =",enroll)
data.head()
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

a) any()

```
print("Enrollment no =",enroll)
np.any(data)
```

```
Enrollment no = 190130107041

True
```

b) all()

```
print("Enrollment no =",enroll)
np.all(2,axis = 0)
```

```
Enrollment no = 190130107041

True
```

c) isnan()

```
In [62]: print(np.isnan(data["age"]))
```

```
0         False
1         False
2         False
3         False
4         False
          ...
566597    False
566598    False
566599    False
566600    False
566601    False
Name: age, Length: 566602, dtype: bool
```

d) isinf()

```
In [60]: print(np.isinf(data["sex"]))
```

```
0         False
1         False
2         False
3         False
4         False
          ...
566597    False
566598    False
566599    False
566600    False
566601    False
Name: sex, Length: 566602, dtype: bool
```

e) isfinite()

```
In [55]: print(np.isfinite(data["sex"]))
```

```
0         True
1         True
2         True
3         True
4         True
          ...
566597    True
566598    True
566599    True
566600    True
566601    True
Name: sex, Length: 566602, dtype: bool
```

f) zeros()

```
print("Enrollment no =",enroll)
print(np.zeros(6))
```

```
Enrollment no = 190130107041
[0. 0. 0. 0. 0. 0.]
```

g) isreal()

```
print("Enrollment no =",enroll)
np.isreal(data)
```

```
Enrollment no = 190130107041

array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
```

h) iscomplex()

```
print("Enrollment no =",enroll)
np.iscomplex(data)
```

```
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False],
```

i)

j) isscaler()

```
In [67]: np.isscalar(data)
Out[67]: False
```

k) less()

```
In [71]: np.less(data['intubed'],data['age'])
Out[71]: 0         False
         1         False
         2          True
         3          True
         4          True
                   ...
         566597    False
         566598    False
         566599     True
         566600    False
         566601    False
         Length: 566602, dtype: bool
```

l) greater()

```
In [72]: np.greater(data['intubed'],data['age'])
```

```
Out[72]: 0          True
         1          True
         2         False
         3         False
         4         False
                   ...
         566597     True
         566598     True
         566599    False
         566600     True
         566601     True
         Length: 566602, dtype: bool
```

m) less_equal()

```
In [73]: np.less_equal(data['intubed'],data['age'])
```

```
Out[73]: 0         False
         1         False
         2          True
         3          True
         4          True
                   ...
         566597    False
         566598    False
         566599     True
         566600    False
         566601    False
         Length: 566602, dtype: bool
```

n) greater_equal()

```
In [74]: np.greater_equal(data['intubed'],data['age'])
```

```
Out[74]: 0          True
         1          True
         2         False
         3         False
         4         False
                   ...
         566597     True
         566598     True
         566599    False
         566600     True
         566601     True
         Length: 566602, dtype: bool
```

15) Develop a program that shows usage of following NumPy library vector functions.

a) arrange()

```
print("Enrollment no =",enroll)
v = np.arange(1,100)
print(v)
```

```
Enrollment no = 190130107041
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
 97 98 99]
```

b) reshape()

```
print("Enrollment no =",enroll)
v = np.arange(0,100).reshape((5,20))
print(v)
```

```
Enrollment no = 190130107041
[[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99]]
```

a

c) linespace()

```
print("Enrollment no =",enroll)
v = np.linspace(4,45,6)
print(v)
```

```
Enrollment no = 190130107041
[ 4.  12.2 20.4 28.6 36.8 45. ]
```

d)randint()

```
In [80]: vector = np.random.randint(0,21,3)
         print(vector)

         [13 20 19]
```

e) dot()

```
In [81]: a1=np.array([2,4,3])
         print(a1)
         a2=np.array([3,4,5])
         print(a2)
         sum=a1.dot(a2)
         print(sum)

         [2 4 3]
         [3 4 5]
         37
```

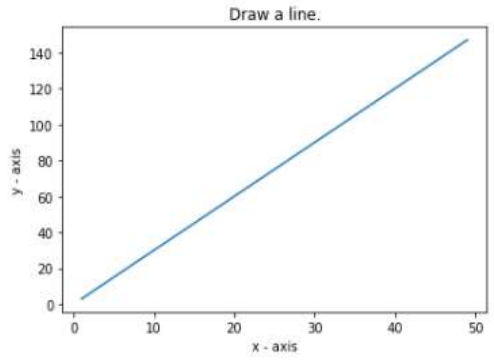
Matplotlib library practicals:

16) Write a program to display below plot using matplotlib library

For Values of X:[1,2,3,...,49], Values of Y (thrice of X):[3,6,9,12,...,144,147]

```
import matplotlib.pyplot as plt

print("Enrollment no =",enroll)

X = range(1, 50)
Y = [value * 3 for value in X]

plt.plot(X, Y)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Draw a line.')

plt.show()

print("Values of X:")
print(*range(1,50))
print("Values of Y (thrice of X):")
print(Y)
```

```
Enrollment no = 190130107041
```




```
Values of X:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49
Values of Y (thrice of X):
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 9
9, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 135, 138, 141, 144, 147]
```

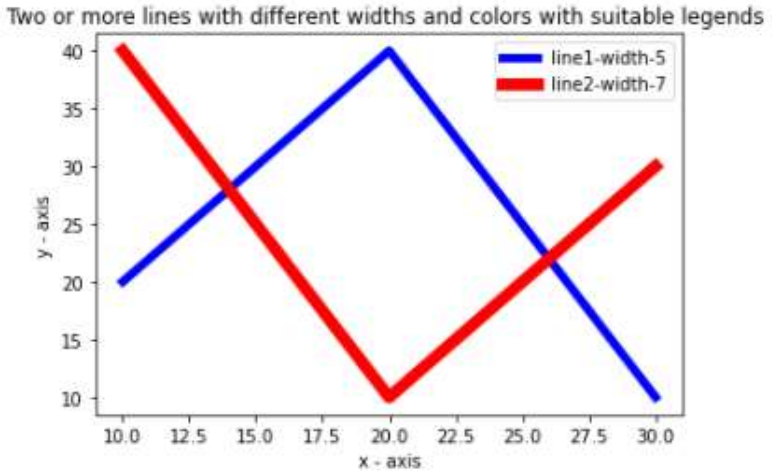
17) Write a program to display below plot using matplotlib library for the point values

x1 = [10,20,30], y1 = [20,40,10]

x2 = [10,20,30], y2 = [40,10,30]

```
print("Enrollment no =",enroll)

x1 = [10,20,30]
y1 = [20,40,10]

x2 = [10,20,30]
y2 = [40,10,30]

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Two or more lines with different widths and colors with suitable legends ')

plt.plot(x1,y1, color='blue', linewidth = 5,  label = 'line1-width-5')
plt.plot(x2,y2, color='red', linewidth = 7,  label = 'line2-width-7')

plt.legend()
plt.show()
```

```
Enrollment no = 190130107041
```



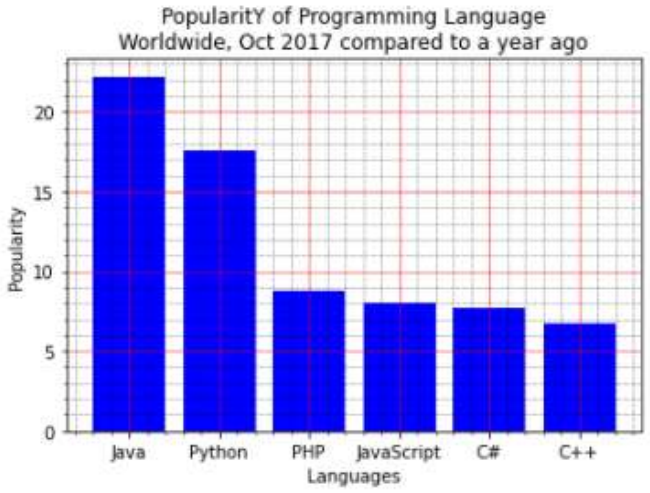
Write a program to display below plot using matplotlib library for the point values

Languages =[‘Java’, ’Python’, ’PHP’, ‘JavaScript’,

‘C#’, ‘C++’] popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

```
print("Enrollment no =",enroll)

x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, popularity, color='blue')


plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")
plt.xticks(x_pos, x)

plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

```
Enrollment no = 190130107041
```



19) Write a program to display below plot using matplotlib library for the point values

languages = [‘Java’, ’Python’, ’PHP’, ’JavaScript’,

’C#’, ’C++’] popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

colours = [#1f77b4, #ff7f0e, #2ca02c, #d62728, #9467bd, #8c564b]

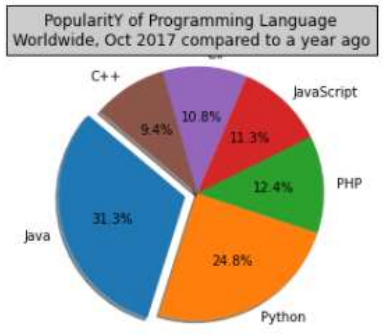
input : -

```
print("Enrollment no =",enroll)

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popuratity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b"]

explode = (0.1, 0, 0, 0,0,0)

plt.pie(popuratity, explode=explode, labels=languages, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a
year ago", bbox={'facecolor':'0.8', 'pad':5})

plt.axis('equal')
plt.show()
```

output : -

```
print("Enrollment no =",enroll)

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popuratity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b"]

explode = (0.1, 0, 0, 0,0,0)

plt.pie(popuratity, explode=explode, labels=languages, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago", bbox={'facecolor':'0.8', 'pad':5

plt.axis('equal')
plt.show()
```

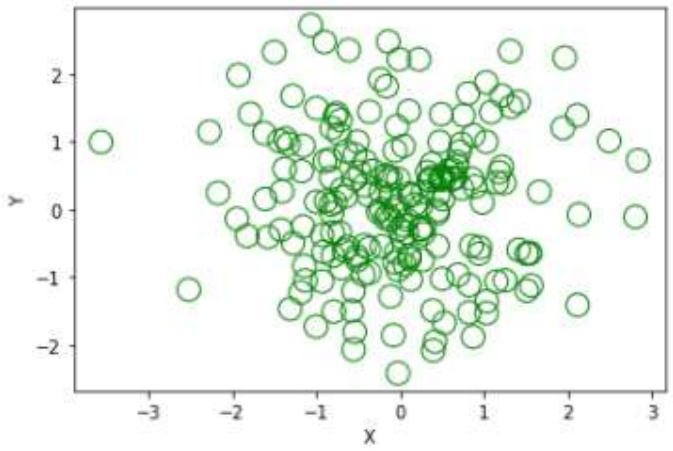
```
Enrollment no = 190130107041
```



20) Write a program to display below bar plot using matplotlib library For 200 random points for both X and Y display scatter plot:

```
print("Enrollment no =",enroll)

from pylab import randn
X = randn(200)
Y = randn(200)
plt.scatter(X,Y, color='r', s=170, facecolors='none', edgecolors='g')

#x = np.random.randn(50)
#y = np.random.randn(50)
#plt.scatter(x, y, s=1, facecolors='none', edgecolors='g')

plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

```
Enrollment no = 190130107041
```



21) Develop a program that reads .csv file from the url:

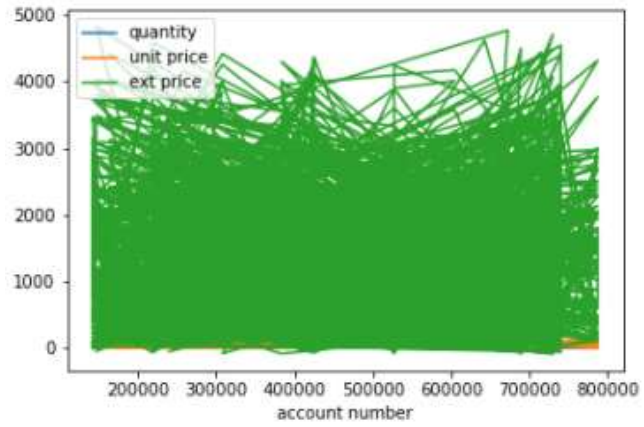
(https://github.com/chris1610/pbpython/blob/master/data/sample-salesv3.xlsx?raw=true) and plot the data of the dataset stored in the .csv file.

Input :-

```
print("Enrollment no =",enroll)

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popuratity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b"]

explode = (0.1, 0, 0, 0,0,0)

plt.pie(popuratity, explode=explode, labels=languages, colors=colors, autopct='%1.1f%%', shadow=True,
startangle=140)
plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago",
bbox={'facecolor':'0.8', 'pad':5})

plt.axis('equal')
plt.show()
```

```
print("Enrollment no =",enroll)

import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_excel('https://github.com/chris1610/pbpython/blob/master/data/sample-salesv3.xlsx?raw=tru
df.plot()
plt.show()
```

```
Enrollment no = 190130107041
```



Scikit Learn practicals:

22) Exercise 1: Language identification

- **Write a text classification pipeline using a custom preprocessor and CharNGramAnalyzer using data from Wikipedia articles as a training set.**
- **Evaluate the performance on some held out test sets.**

```
import sys
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.pipeline import Pipeline
from sklearn.datasets import load_files
from sklearn.model_selection import train_test_split
from sklearn import metrics
languages_data_folder = "short_paragraphs/"
dataset = load_files(languages_data_folder)
docs_train, docs_test, y_train, y_test = train_test_split(
 dataset.data, dataset.target, test_size=0.5)
vectorizer = TfidfVectorizer(ngram_range=(1, 3), analyzer='char',
 use_idf=False)
clf = Pipeline([
 ('vec', vectorizer),
 ('clf', Perceptron()),
])
clf.fit(docs_train, y_train)
y_predicted = clf.predict(docs_test)
sentences = [
 'This is a language detection test.',
 'Ceci est un test de d\xe9tection de la langue.',
 'Dies ist ein Test, um die Sprache zu erkennen.',
]
predicted = clf.predict(sentences)
print()
print()
for s, p in zip(sentences, predicted):
 print('The language of "%s" is "%s"' % (s, dataset.target_names[p]))
```

```
The language of "This is a language detection test." is "en"
The language of "Ceci est un test de détection de la langue." is "fr"
The language of "Dies ist ein Test, um die Sprache zu erkennen." is "de"
```

23) Exercise 2: Sentiment Analysis on movie reviews

- **Write a text classification pipeline to classify movie reviews as either positive or negative.**
- **Find a good set of parameters using grid search.**
- **Evaluate the performance on a held out test set.**

```
import numpy as np
import pandas as pd
import sklearn
class Review:
 def get_sentiment(self):
 if self.tag == "pos":
 return "POSITIVE"
 else:
 return "NEGATIVE"
 def __init__(self, text, tag):
 self.text = text
 self.tag = tag
 self.sentiment = self.get_sentiment()
data = pd.read_csv('movie_review.csv')
reviews = []
for i in range(len(data)):
 reviews.append(Review(data['text'][i], data['tag'][i]))
from sklearn.model_selection import train_test_split
training, test = train_test_split(reviews, test_size=0.33,random_state=42)
train_x = [x.text for x in training]
train_y = [x.sentiment for x in training]
test_x = [x.text for x in test]
test_y = [x.sentiment for x in test]
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
train_x_vectors = vectorizer.fit_transform(train_x)
```

```
from sklearn.feature_extraction.text import TfidfTransformer
tf_transformer = TfidfTransformer()
train_x_tf = tf_transformer.fit_transform(train_x_vectors)
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(train_x_tf, train_y)
x_new_counts = vectorizer.transform(test_x)
x_new_tf = tf_transformer.transform(x_new_counts)
predicted = clf.predict(x_new_tf)
from sklearn.pipeline import Pipeline
text_clf = Pipeline([
 ('vect',CountVectorizer()),
 ('tfidf', TfidfTransformer()),
 ('clf',MultinomialNB()),
])
text_clf.fit(train_x, train_y)
from sklearn.model_selection import GridSearchCV
parameters = {
 'vect__ngram_range': [(1, 1), (1, 2), (2,4)],
 'tfidf__use_idf': (True, False),
 'clf__alpha': (1e-2, 1e-3,1e-4),
}
gs_clf = GridSearchCV(text_clf, parameters, cv=5, n_jobs=-1)
gs_clf = gs_clf.fit(train_x,train_y)
predicted = gs_clf.predict(test_x)
print("Mean score: ",np.mean(predicted == test_y))
check_review = ["Worst movie ever seen",]
predicted = gs_clf.predict(check_review)
print(predicted)
```

```
Mean score:  0.6994100571214533
['NEGATIVE']
```

24) CLI text classification utility

Using the results of the previous exercises and the cPickle module of the standard library, write a command line utility that detects the language of some text provided on stdin and estimates the polarity (positive or negative) if the text is

written in English.

Bonus point if the utility is able to give a confidence level for its predictions.

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.metrics import confusion_matrix
class Review:
 def get_sentiment(self):
 if self.tag == "pos":
 return "POSITIVE"
 else:
 return "NEGATIVE"


 def __init__(self, text, tag):
 self.text = text
 self.tag = tag
 self.sentiment = self.get_sentiment()

data = pd.read_csv('movie_review.csv')
reviews = []
for i in range(len(data)):
 reviews.append(Review(data['text'][i], data['tag'][i]))
from sklearn.model_selection import train_test_split
training, test = train_test_split(reviews, test_size=0.33,random_state=42)
train_x = [x.text for x in training]
train_y = [x.sentiment for x in training]
test_x = [x.text for x in test]
test_y = [x.sentiment for x in test]
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
train_x_vectors = vectorizer.fit_transform(train_x)
from sklearn.feature_extraction.text import TfidfTransformer
tf_transformer = TfidfTransformer()
train_x_tf = tf_transformer.fit_transform(train_x_vectors)
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(train_x_tf, train_y)
x_new_counts = vectorizer.transform(test_x)
x_new_tf = tf_transformer.transform(x_new_counts)
predicted = clf.predict(x_new_tf)
from sklearn.pipeline import Pipeline
text_clf = Pipeline([
 ('vect',CountVectorizer()),
 ('tfidf', TfidfTransformer()),
 ('clf',MultinomialNB()),
])
text_clf.fit(train_x, train_y)
predict = text_clf.predict(test_x)
from sklearn.model_selection import GridSearchCV
parameters = {
 'vect__ngram_range': [(1, 1), (1, 2), (2,4)],
 'tfidf__use_idf': (True, False),
 'clf__alpha': (1e-2, 1e-3,1e-4),
}
gs_clf = GridSearchCV(text_clf, parameters, cv=5, n_jobs=-1)
gs_clf = gs_clf.fit(train_x,train_y)
predicted = gs_clf.predict(test_x)
print(confusion_matrix(predicted, test_y))
print('Mean score:', np.mean(predicted == test_y))
import sys
data = input(sys.argv)
predicted = gs_clf.predict([data])
print(predicted)
```

```
[[7155 3067]
 [3353 7783]]
Mean score: 0.6994100571214533
['C:\\Users\\milind\\anaconda3\\lib\\site-packages\\ipykernel_launcher.py', '-f', 'C:\\Users\\milind\\AppData\\Roaming\\jupyter
\\runtime\\kernel-31a0fe50-4137-42fe-b1b4-1db546f91665.json']Hello, I am Milind .This film is beautiful
['POSITIVE']
```